PROJECT DOCUMENTATION

Overview-

* Transfer files from S3 bucket to remote server using SFTP using SFTP Protocol.
* Files need to be sent daily at a fixed time.
* Files need to be sent incrementally i.e. the same file should not be sent again.
* Log needs to be maintained of files transferred via SFTP.

In our use case we will be using Amazon Linux EC2 instance as our remote server.

Initial Setup

1. Creating the S3 bucket.

We will be requiring paramiko library for executing linux commands over remote instances via SSH. For this we will create an AWS lambda layer .

* On the EC2 instance, install necessary packages.

sudo yum install -y python3-pip zip

* Make a directory structure as shown below

mkdir -p python/lib/python3.x/site-packages/

where x represents the python version that will be used in lambda function.

* Inside this folder structure, install paramiko library

Pip3 install paramiko -t .

* Zip the folder

zip -r9 paramiko-layer.zip python

* Transfer the file to your S3 Bucket.

aws s3 cp paramiko-layer.zip s3://your-s3-bucket-name/

Important note- Ensure that the cryptography and bcrypt versions match the AWS Lambda environment.

It is advisable to create python folder inside python virtual environment. This will help avoiding the conflicts.

1. Create IAM roles.

We need to create two IAM roles for this task, One for lambda function and another for SFTP server.

IAM role for Lambda Function.

* AmazonBasicLambdaExecutionRole
* AmazonS3FullAccess

IAM role for EC2 Instance

* AmazonS3WriteAccess

1. Inside the S3 bucket create a folder which would contain the private key of the remain instance.
2. Create a lambda layer for lambda function.

* In functions create a lambda layer and choose the zip file from S3 bucket.
* Create the main lambda function and add this layer to the lambda function.

1. Create the main lambda function in python.
2. Create folders in the S3 bucket for testing folders. These are the folders which need to be backed up in S3 bucket.
3. Add some files to these folders.
4. Test the code.

Lambda function code in python.

import paramiko

import boto3

import io

import datetime

import os

def lambda\_handler(event, context):

    # Initialize S3 client

    S3Client = boto3.client('s3')

    # Define S3 bucket and folder names

    bucket\_name = 'sftp-protocol-bucket'

    folders = ['folder1/', 'folder2/']  # List of folders to process

    # Download the PEM key for the remote instance

    S3Client.download\_file(bucket\_name, 'private-key/smtp.pem', '/tmp/remote.pem')

    pem\_key = paramiko.RSAKey.from\_private\_key\_file("/tmp/remote.pem")

    # Create a new SSH client

    SSHClient = paramiko.SSHClient()

    SSHClient.set\_missing\_host\_key\_policy(paramiko.AutoAddPolicy())

    # Remote instance details

    remote\_host = "141.148.215.102"

    remote\_user = "opc"

    # Connect to the remote instance

    SSHClient.connect(hostname=remote\_host, username=remote\_user, pkey=pem\_key)

    print("Connected to: " + remote\_host)

    # Open SFTP session

    SFTPClient = SSHClient.open\_sftp()

    total\_files\_transferred = 0

    log\_entries = []

    timestamp = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')

    for folder\_name in folders:

        # List objects in the specified S3 folder

        objects = S3Client.list\_objects\_v2(Bucket=bucket\_name, Prefix=folder\_name)

        folder\_files\_transferred = 0

        # Ensure the remote folder exists

        remote\_folder = f'/home/opc/uploads/{folder\_name.strip("/")}'

        try:

            SFTPClient.mkdir(remote\_folder)

            print(f"Created remote folder: {remote\_folder}")

        except IOError:

            print(f"Remote folder already exists: {remote\_folder}")

        # Transfer files from S3 to the remote instance

        for obj in objects.get('Contents', []):

            if obj['Key'].endswith('/'):

                # Skip if it's a folder (common in S3)

                continue

            file\_key = obj['Key']

            filename = file\_key.split('/')[-1]

            remote\_file = os.path.join(remote\_folder, filename)  # Adjust target directory on the remote instance

            # Stream the file from S3 and upload to remote instance

            s3\_object = S3Client.get\_object(Bucket=bucket\_name, Key=file\_key)

            file\_size = s3\_object['ContentLength']

            file\_stream = io.BytesIO(s3\_object['Body'].read())

            with SFTPClient.file(remote\_file, 'wb') as remote\_file\_obj:

                remote\_file\_obj.write(file\_stream.read())

            log\_entries.append(f"{filename} ({file\_size} bytes) transferred from {folder\_name} to {remote\_file} on {timestamp}")

            folder\_files\_transferred += 1

            print(f"Transferred {filename} ({file\_size} bytes) from {folder\_name} to {remote\_file}")

        total\_files\_transferred += folder\_files\_transferred

        print(f"Transferred {folder\_files\_transferred} file(s) from {folder\_name}")

    # Close SFTP and SSH connections

    SFTPClient.close()

    SSHClient.close()

    # Create log file content

    log\_content = '\n'.join(log\_entries)

    # Create logs directory in S3 bucket

    log\_filename = datetime.datetime.now().strftime('logs%Y-%m-%d.txt')

    log\_file\_key = f'logs/{log\_filename}'

    # Upload log file to S3 bucket

    S3Client.put\_object(Bucket=bucket\_name, Key=log\_file\_key, Body=log\_content)

    return f"{total\_files\_transferred} file(s) have been transferred to the remote instance and logged in S3."

Code Explanation-

**Import Necessary Libraries**: The function imports required libraries (paramiko for SSH/SFTP, boto3 for interacting with AWS services, io for handling streams, datetime for timestamps, and os for file path manipulations).

**Initialize S3 Client**: boto3 is used to create an S3 client to interact with the S3 service.

**Define S3 Bucket and Folder Names**: The S3 bucket name is set to 'sftp-protocol-bucket', and a list of folder names (['folder1/', 'folder2/']) is defined to indicate which folders to process.

**Download the PEM Key**: The function downloads the private key (smtp.pem) from the S3 bucket to a temporary location (/tmp/remote.pem).

**Load PEM Key**: The PEM key is loaded using paramiko.RSAKey.from\_private\_key\_file.

**Create SSH Client**: An SSH client is created and configured to automatically add the server's host key if it's missing.

**Remote Instance Details**: The remote server's hostname and username are specified.

**Connect to Remote Server**: The SSH client connects to the remote server using the hostname, username, and private key.

**Open SFTP Session**: An SFTP session is opened through the SSH connection.

**Initialize Counters and Log Entries**: Variables are initialized to keep track of the total files transferred and log entries. A timestamp is also generated for logging purposes.

**Process Each Folder**: For each folder in the list:

* **List Objects in S3 Folder**: The contents of the folder are listed using list\_objects\_v2.
* **Ensure Remote Folder Exists**: The function tries to create the corresponding folder on the remote server. If it already exists, it continues without error.
* **Transfer Files**: Each file in the S3 folder (skipping subfolders) is streamed and uploaded to the remote server:
  + The file is retrieved from S3.
  + Its content is read into a stream.
  + The stream is written to the remote server using SFTP.
  + A log entry is created for each file transferred, noting the filename, size, source folder, destination path, and timestamp.
  + Counters for the number of files transferred are updated.

**Close Connections**: The SFTP and SSH connections are closed.

**Create Log File Content**: The log entries are joined into a single string.

**Upload Log File to S3**: The log file is named with the current date and uploaded to the S3 bucket in the logs folder.